Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anže Nunar

Algoritem COMP128 na kartici SIM

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in matematika

Mentorica: doc. dr. Arjana Žitnik

Ljubljana, 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Algoritem COMP128 na kartici SIM

Tematika naloge:

Kartica SIM v mobilnem telefonu med drugim sodeluje pri avtentikaciji v omrežje in generiranju sejnih ključev, ki se nato uporabljajo za šifriranje komunikacij po mobilnem omrežju. Podatki za avtentikacijo in sejni ključ se izračunajo z eno od različic algoritma COMP128.

V diplomskem delu opišite vlogo kartice SIM v omrežju GSM. Podrobno opišite algoritem COMP128-1 in analizirajte njegovo varnost. Predstavite tudi koncepte iz kriptografije, ki jih pri tem uporabljate.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Anže Nunar sem avtor diplomskega dela z naslovom:

Algoritem COMP128 na kartici SIM

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Arjane Žitnik,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 18. septembra 2015 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AES	advanced encryption standard	napredni šifrirni standard
AuC	authentication center	center za avtentikacijo
BSC	base station controller	nadzornik
BTS	base transceiving station	bazne sprejemno-oddajne postaje
CKSN	ciphering key sequence number	šifrirna kodna sekvenčna številka
GSM	global system for mobile communications	globalni sistem mobilnih komunikacij
HLR	home location register	domači register
ICCID	integrated circuit card identifier	identifikacijska kartica z integriranim vezjem
IMSI	international mobile subscriber identity	mednarodna identiteta mobilnega naročnika
LAI	location area identity	identiteta lokacijskega območja
MSC	mobile switching center	glavni preklopni center
PIN	personal identification number	osebna identifikacijska številka
PUK	PIN unblocking code	koda za deblokiranje
SIM	subscriber identity module	enota za identiteto naročnika
TMSI	temporary mobile subscriber identity	začasna številka identitete naročnika
VLR	visitor location register	register obiskovalcev

Povzetek

Kartica SIM igra pomembno vlogo pri mobilni komunikaciji. Na kartici SIM je shranjen zasebni ključ uporabnika. Do njega dostopa algoritem COMP128, ki omogoča avtentikacijo uporabnika v mobilno omrežje in generiranje sejnih ključev za šifriranje komunikacij.

V diplomskem delu podrobno opišemo algoritem COMP128. Napad nanj izvedemo z grobo silo. Pri tem izkoristimo slabosti v strukturi algoritma COMP128 in paradoks rojstnega dne, kar omogoča rekonstrukcijo zasebnega ključa v realnem času. Napad smo tudi implementirali in preizkusili. V posebnem poglavju predstavimo uporabljene koncepte iz kriptografije. To so avtentikacija, zgoščevalne funkcije in paradoks rojstnega dne.

Ključne besede: algoritem COMP128, avtentikacija, kartica SIM, kriptografija, napad z grobo silo, protokol GSM, paradoks rojstnega dne.

Abstract

SIM card has an important role in mobile communications. It contains the user's private key, which is used in COMP128 algorithm for user authentication to the mobile network and generating session keys to encrypt communications.

In this thesis a detailed description of COMP128 algorithm is presented. A brute force attack on the algorithm is described, which exploits a weakness in the structure of COMP128 algorithm, and is based on birthday paradox. As a result the user's private key can be obtained in real time. The attack on COMP128 algorithm was also implemented and tested. A separate chapter provides the necessary cryptographic concepts such as authentication, hash functions and birthday paradox.

Keywords: COMP128 algorithm, authentication, SIM card, cryptography, brute-force attack, GSM protocol, birthday paradox.

Poglavje 1

Uvod

Množičen razvoj mobilne telefonije se je začel pred dobrima dvema desetletjema. Takrat so se pojavile tudi prve kartice SIM in mobilni telefoni, ki so uporabljali omrežje GSM. Danes mobilne telefone uporablja že približno štiri milijarde ljudi, kar predstavlja skoraj polovico prebivalstva. Čeprav bi zaradi hitrega napredka tehnologije pričakovali, da današnji pametni telefoni uporabljajo novo tehnologijo, to večinoma ni res. Za svoje delovanje še vedno potrebujejo skoraj enako kartico SIM kot na začetku. Žal pa je sicer dobrodošel tehnološki napredek prinesel tudi nove izzive, s katerimi se sooča tudi mobilna telefonija. Eden izmed njih je vsekakor zagotavljanje varnih komunikacij in preprečevanje prisluškovanja. Kljub sprejetju novih standardov se je izkazalo, da se še vedno uporabljajo stare tehnologije, ki niso varne in omogočajo zlorabe.

Za uporabo omrežja GSM potrebujemo kartico SIM, ki jo vstavimo v mobilni telefon. Gre za pametno kartico, na kateri so shranjeni podatki, ki enolično določajo uporabnika, podobno kot osebna izkaznica. Najpomembnejši med njimi je zagotovo zasebni ključ K_i. Skupaj z algoritmom COMP128 omogoča avtentikacijo v omrežje in izračun sejnih ključev. Sejni ključi se uporabljajo pri šifriranju komunikacij po mobilnem omrežju. Njihova pogosta menjava pa otežuje delo napadalcem, ki nam želijo prisluškovati.

V ozadju delovanja algoritma COMP128 se skrivajo zgoščevalne funk-

cije. Zgoščevalne funkcije predstavljajo pomembno orodje v kriptografiji, matematični vedi, ki se ukvarja z varno komunikacijo po nezaščitenih komunikacijskih kanalih. Tudi kriptografija je z razmahom računalniških omrežij konec 70. let doživela svoj razcvet. Zaradi tehnološkega napredka pa so nekateri problemi, ki so včasih veljali za težke, postali enostavno rešljivi. Eden izmed njih je zagotovo tudi napad na algoritem COMP128. Napad nanj je bilo ob začetku njegovo uporabe mogoče narediti le z drago opremo, danes pa to zmore vsak osebni računalnik. Opis algoritma COMP128 in napad nanj je osrednja tema tega diplomskega dela.

Na kratko si poglejmo zgradbo diplomskega dela. Po uvodnem poglavju v poglavju 2 predstavimo teoretične osnove, pomembne za diplomsko delo, s področja kriptologije. Opišemo kompresijske in zgoščevalne funkcije, ki so temelj algoritma COMP128. Poglavje zaključimo s predstavitvijo protokola za avtentikacijo med strežnikom in odjemalcem. To poglavje je večinoma povzeto po knjigah [9] in [11]. Sledi poglavje 3, ki je namenjeno kartici SIM. Najprej podrobneje predstavimo kakšni podatki se na njej shranjujejo, nato pa pojasnimo še njeno vlogo v omrežju GSM. To je sodelovanje pri avtentikaciji v omrežje in generiranju sejnih ključev. V poglavju 4 podrobneje obravnavamo algoritem COMP128 ter opišemo napad nanj. Napad smo implementirali in ga izvedli na računalniku brez uporabe kartice SIM. V zadnjem poglavju, poglavju 5, povzamemo rezultate diplomskega dela in opozorimo na nekatere druge pomanjkljivosti omrežja, ki se pojavljajo v zvezi z avtentikacijo.

Poglavje 2

Kriptologija

Beseda *kriptologija* je grškega izvora in je sestavljena iz besed *kryptos* in *logos*, kar v prevodu pomeni skrita beseda. Kriptologijo delimo na dve, med seboj povezani področji, kriptografijo in kriptoanalizo. *Kriptografija* se ukvarja z načrtovanjem šifer in protokolov, *kriptoanaliza* pa z njihovim razbijanjem.

Cilji kriptografije, sprva sinonima za šifriranje podatkov, so se z razmahom računalniških omrežij razširili. V času druge svetovne vojne in prvih računalnikov so šifre, ki jih bomo imenovali kriptosistemi, postajale vse bolj zapletene, njihova uporaba pa vse bolj pogosta. Med glavne cilje sodobne kriptografije sodijo:

- *zaupnost* (angl. confidentiality): predstavlja ohranjanje tajnosti podatkov z namenom, da lahko do želenih podatkov dostopa le pooblaščena oseba,
- *celovitost podatkov* (angl. data integrity): predstavlja zaščito podatkov pred nedovoljenimi spremembami,
- *avtentikacija* (angl. authentication): preverjanje identite,
- *preprečevanje tajenja* (angl. non-repudiation): nezmožnost zanikanja že storjenih dejanj.

Kriptolog A. Kerckhoffs je v svojih člankih *La Cryptographie Militaire* [7, 8] naštel šest načel, kako naj bodo pripravljeni varni kriptosistemi. Trdil

je, da je kriptosistem lahko javen in da ga to ne sme ogrožati. Četudi je kriptosistem matematično zlomljiv, to v praksi ne sme veljati. Ključ si je potrebno zapomniti, ne da bi si ga bilo potrebno zapisati, za uporabo kriptosistema pa ne sme biti potrebno delo več ljudi. Biti mora dovolj enostaven, da za njegovo uporabo ni potreben miselni napor in upoštevanje velikega števila pravil. *Kerchoffsovo načelo* trdi, da je kriptosistem varen, tudi če je o njem znano vse, razen šifrirnega ključa. Tega načela se v kriptografiji držimo še danes.

Zaradi široke uporabe kriptografije se je vse bolj razvijala tudi kriptoanaliza. Cilji napadalca so različni, od dešifriranja zašifriranih sporočil, ki jih imenujemo tudi kriptogram ali tajnopis, do razkritja zasebnih ključev.

Pri kriptoanalizi ločimo *pasivne* in *aktivne* napade. Pri pasivnih napadih napadalec lahko zgolj prisluškuje, sporočil pa ne more spreminjati. Med takšne napade spadata *napad z golim kriptogramom* (angl. ciphertext only attack), kjer napadalec pozna enega ali več kriptogramov ter *napad z znanim besedilom* (angl. known plaintext attack), kjer napadalec pozna enega ali več parov (besedilo, kriptogram). Primer takšnih napadov so napadi z izčrpnim pregledovanjem vseh ključev, različne statistične metode (na primer frekvenčna analiza), izkoriščanje algebraične strukture kriptosistema za izračun ključa in podobno.

Aktivni napadi so napadi, pri katerih napadalec lahko prestreza, spreminja, dodaja ter briše sporočila. Mednje spadata *napad z izbranim besedilom* (angl. chosen plaintext attack), pri katerem napadalec dobi začasen dostop do kodirnega postopka, kjer lahko generira pare (besedilo, kriptogram) za izbrana besedila in preizkuša domneve ter *napad z izbranim kriptogramom* (angl. chosen ciphertext attack), kjer ima napadalec začasen dostop do dekodirnega postopka ter lahko generira pare (besedilo, kriptogram) za izbrane kriptograme, vendar ne pozna dekodirnega ključa.

Ker ne želimo, da bi postali tarča napadalca, moramo poskrbeti za ustrezno zaščito. Poleg šifriranja sporočil z dobrim kriptosistemom moramo poskrbeti tudi za varnost pri prenosu sporočil. Pred začetkom pošiljanja sporočil

se želimo prepričati, da je na drugi strani res prava oseba. To preverimo v postopku avtentikacije.

V naslednjih razdelkih na kratko predstavimo koncepte iz kriptografije, ki jih bomo potrebovali v nadaljevanju.

2.1 Šifriranje

Šifriranje sporočil je pretvorba izvornega sporočila, imenovanega *besedilo* (tudi čistopis, angl. plaintext) z uporabo algoritma v obliko, da ga nepooblaščene osebe ne morejo razumeti. Šifrirano sporočilo imenujemo *kriptogram* (tudi tajnopis, angl. ciphertext). Uporaba šifriranja je bila sprva razširjena v diplomaciji, oboroženih silah ter obveščevalnih dejavnostih, v zadnjem času pa postaja vse bolj pogosta tudi v poslovnem svetu ter za šifriranje osebnih podatkov.

Zapišimo še, kako šifriranje formalno predstavimo s pomočjo *kriptosistema* ali *šifre*. Kriptosistem $\mathcal{S}$ je peterka $(\mathcal{B}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, za katero velja:

- $\mathcal{B}$ predstavlja končno množico *besedil* (angl. plaintext),
- $\mathcal{C}$ predstavlja končno množico *kriptogramov* (angl. ciphertext),
- $\mathcal{K}$ predstavlja končno množico *ključev*,
- $\mathcal{E}$ predstavlja družino *kodirnih funkcij*, za katere velja:

$$\mathcal{E} = \{E_k : \mathcal{B} \to \mathcal{C} : k \in \mathcal{K}\},$$

- $\mathcal{D}$ predstavlja družino *dekodnih funkcij*, za katere velja:

$$\mathcal{D} = \{D_k : \mathcal{C} \to \mathcal{B} : k \in \mathcal{K}\},$$

- za vsak $e \in \mathcal{K}$ obstaja $d \in \mathcal{K}$, da velja $D_d(E_e(b)) = b$, za vsak $b \in \mathcal{B}$.

Trditev 2.0.1 *Za vsak $k \in K$ je kodirna funkcija E_k injektivna.*

Dokaz. Izberimo si poljubni besedili: b_1 in $b_2 \in \mathcal{B}$, za kateri velja:

$$E_k(b_1) = E_k(b_2).$$

Naj bo d dekodirni ključ, ki ustreza zadnji lastnosti kriptosistema velja:

$$D_d(E_k(b_1)) = D_d(E_k(b_2)),$$

od koder sledi:

$$b_1 = b_2.$$

□

Če kodirna funkcija ne bi bila injektivna, dekodiranje ne bi bilo mogoče na nedvoumen način.

Danes se za šifriranje največ uporablja tudi *napredni šifrirni sistem* AES (angl. advanced encryption standard). Kljub številnim poizkusom do danes nanj še ni bilo izvedenega uspešnega napada, zato še vedno velja za varnega.

2.2 Kompresijske in zgoščevalne funkcije

Kompresijska funkcija (angl. compression function) f sporočilu, dolžine $n+r$ priredi kratek *povzetek* (angl. message digest, hash) fiksne dolžine n:

$$f : \{0,1\}^{r+n} \to \{0,1\}^n.$$

Zgoščevalna funkcija (angl. hash function) h sporočilu poljubne dolžine priredi kratek *povzetek* (angl. message digest, hash) fiksne dolžine:

$$h : \{0,1\}^* \to \{0,1\}^n.$$

sporočilo	povzetek
00 00 00 00	7E 07 1F D9 B0 23 ED 8F 18 45 8A 73 61 3A 08 34 F6 22 0B D5 CC 50 35 7B A3 49 3C 60 40 A9 EA 8C
00 00 00 01	AE 5C E1 62 88 8E E3 EB E9 74 97 6C AC 5A B9 4A 3F 55 04 9F 85 15 88 48 83 D5 79 FB 3F A3 78 D2

Tabela 2.1: Dva povzetka kriptografske zgoščevalne funkcije SHA-256.

Zgoščevalne funkcije morajo biti enostavne in morajo omogočati hitro računanje povzetka. V kriptografiji zgoščevalne funkcije igrajo pomembno vlogo. Uporabljajo se na primer za:

- zagotavljanje celovitosti podatkov,
- digitalne podpise, kjer podpišemo le povzetek sporočila,
- hranjenje gesel, kjer namesto gesel hranimo njihove povzetke.

Kriptografska zgoščevalna funkcija je običajno sestavljena tako, da iterativno kliče kompresijske funkcije. Da so kriptografske zgoščevalne funkcije varne, morajo imeti še nekatere dobre lastnosti:

- *Naključnost.* Povzetek dveh sporočil, ki se razlikujeta le na enem mestu, mora izgledati kot neodvisno izbrani naključni števili. V tabeli 2.1 je prikazana uporaba kriptografske zgoščevalne funkcije SHA-256 za dve števili, ki se razlikujeta le na enem mestu. Opazimo, da sta njuna povzetka povsem različna.
- *Odpornost praslik.* Za poljuben povzetek mora biti računsko nemogoče poiskati sporočilo, kar pomeni, da morajo biti kriptografske zgoščevalne funkcije *enosmerne funkcije.*
- *Odpornost drugih praslik.* Za dano sporočilo je računsko nemogoče najti drugo sporočilo, ki ima enak povzetek.

- *Odpornost na trke. Trk* je par različnih sporočil x in x' z enakim povzetkom. Ker zgoščevalna funkcija vedno slika iz večje množice v manjšo, po Dirichletovem načelu vedno obstaja sporočilo z istim povzetkom. Želena lastnost je torej, da je trke računsko nemogoče poiskati.

Zgornje lastnosti lahko dosežemo z *razpršitvijo* (angl. diffusion) in *zmedo* (angl. confusion). Razpršitev pomeni, da mora biti vsak znak povzetka odvisen od vseh znakov sporočila, zmeda pa zahteva čim bolj zapleteno zvezo med povzetkom in ključem.

2.2.1 Paradoks rojstnega dne

V teoriji verjetnosti se paradoks rojstnega dne nanaša na verjetnost, kjer za dovolj veliko množico k naključnih ljudi lahko trdimo, da imata vsaj dva rojstni dan na isti dan. Z uporabo Dirichletovega načela lahko trdimo z verjetnostjo 1, da v množici 366 ljudi obstaja vsaj en par, ki ima rojstni dan na isti dan (zaradi enostavnejšega računanja privzemimo, da ima leto 365 dni). V množici 70 ljudi lahko to še vedno trdimo z verjetnostjo 0.999. Verjetnosti za manjše število oseb so predstavljene v tabeli 2.2.

Funkcijo, ki slika ljudi v njihove rojstne dneve, lahko gledamo kot zgoščevalno funkcijo. Najdeni par, ki ima rojstni dan na isti dan, nam predstavlja trk. Opazimo, da je v množici 23 ljudi verjetnost, da ima vsaj en par rojstni dan na isti dan, približno 0.507. To nam potrdi kratek izračun, v katerem izračunamo verjetnost nasprotnega dogodka:

$$1 - \frac{365}{365} \cdot \frac{364}{365} \cdot \frac{363}{365} \cdots \frac{343}{365} \approx 0.507.$$

Ocenimo verjetnost, da v n dneh noben par izmed k oseb nima rojstnega dne na isti dan. Pri tem uporabimo oceno $1 - x \leq e^{-x}$:

$$\sum_{i=0}^{k-1} \left(\frac{n-i}{n} \right) = \sum_{i=1}^{k-1} \left(1 - \frac{i}{n} \right) \leq \sum_{i=1}^{k-1} e^{\frac{-i}{n}} = e^{\frac{-k(k-1)}{2n}}.$$

število oseb	verjetnost
1	0
5	0.027
10	0.117
20	0.411
23	0.507
30	0.706
40	0.891
50	0.970
60	0.994
70	0.999

Tabela 2.2: Verjetnost, da imata vsaj dve osebi rojstni dan hkrati.

Torej je verjetnost trka enaka vsaj

$$1 - e^{\frac{-k(k-1)}{2n}} \approx 1 - e^{\frac{-k^2}{2n}}.$$

Odtod sledi, da je za $k \geq \sqrt{2n \ln 2} \approx 1,17\sqrt{n}$ verjetnost trka večja od 0.5. Torej pri izračunu povzetkov za $\sqrt{2}^n = 2^{\frac{n}{2}}$ sporočil, bomo z verjetnostjo približno 0.5 našli trk. Za kriptografske zgoščevalne funkcije si torej želimo, da je v primeru, ko napadalec lahko spreminja obe sporočili, dolžina povzetka n dovolj velika, da je računsko nemogoče izvesti napad z grobo silo, ki preveri $2^{\frac{n}{2}}$ sporočil.

2.2.2 Iskanje trkov

V tem podrazdelku bomo opisali, kako za dano zgoščevalno funkcijo

$$h = \{0,1\}^* \to \{0,1\}^n$$

poiščemo trk z grobo silo. Pri tem bomo izkoristili paradoks rojstnega dne. Iščemo torej dve različni sporočili, ki imata enak povzetek. Obstoj trka nam

znova zagotavlja Dirichletovo načelo, ker zgoščevalna funkcija slika iz večje množice v manjšo.

Naključno izbiramo števila in jih vstavljamo v zgoščeno tabelo, kjer kot ključ za število x uporabimo njegov povzetek $h(x)$. Vsakič preverimo, če je povzetek izbranega števila že v zgoščeni tabeli. V kolikor ga najdemo v zgoščeni tabeli, poiščemo še število, ki ustreza temu povzetku. S tem smo našli trk in iskanje zaključimo. Če trka ne najdemo, potem v zgoščeno tabelo dodamo trenutno število ter njegov povzetek. Pri tem uporabimo lastnost zgoščene tabele, da se operacije iskanja, vstavljanja ter brisanja izvedejo v konstantnem času. Delovanje nam prikazuje algoritem 1.

Algoritem mora v najslabšem primeru generirati $2^{n/2}+1$ naključnih števil. Pričakovana časovna zahtevnost je zaradi upoštevanja paradoksa rojstnega dne $\mathcal{O}(2^{n/2})$.

```
vhod : /
izhod: {x,x'} : h(x) = h(x')
while true do
    x = rand();
    if h(x) in hash table T then
        x' = T.get(h(x));
        return (x, x');
    else
        T.add (h(x), x)
    end
end
```

Algoritem 1: Iskanje trka z grobo silo.

2.3 Avtentikacija

V tem razdelku bomo predstavili postopek avtentikacije, ki je lahko enostranska ali dvostranska. Komunikacijo med strežnikom in odjemalcem je

mogoče prestreči, kar imenujemo napad srednjega moža. Razdelek je povzet po knjigah [9, 11].

Postopek, v katerem odjemalec prepriča strežnik, da je res to, za kar se predstavlja, imenujemo *avtentikacija* (angl. authentication). Primer preprostega načina avtentikacije je vpis gesla. Odjemalec mora poznati geslo, po njegovem vpisu pa strežnik preveri, če je geslo pravilno. S tem strežnik predvideva, da je odjemalec pooblaščen za dostop do informacij. Seveda lahko zahtevamo tudi obratno preverjanje, da odjemalec preveri, če je res strežnik pravi.

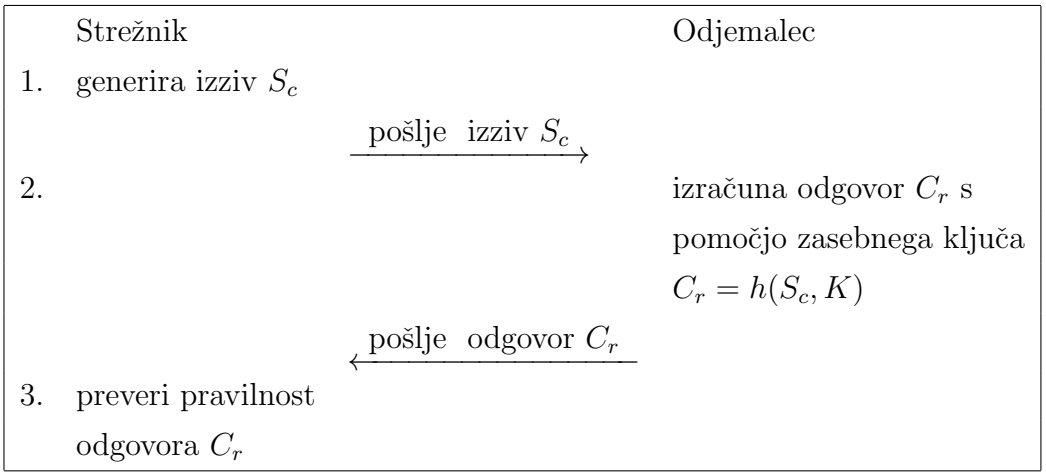
Za avtentikacijo lahko uporabimo:

- nekaj, kar *imamo* (na primer ključ ali kartica),
- nekaj, kar *vemo* (na primer geslo),
- nekaj, kar *smo* (biometrične lastnosti, na primer prstni odtisi).

Pri uporabi le ene od naštetih opcij ima napadalec lažje delo. S krajo ključa ali kartice, poskušanjem gesla ali prestrezanjem prometa, se napadalec lahko uspešno avtenticira v sistem. Zato se ponavadi v postopku avtentikacije uporablja več načinov.

Primer enostavnega postopka, ki za avtentikacijo potrebuje dve od naštetih opcij, je uporaba bankomata. Uporabnik mora za uspešno prijavo v bankomat imeti fizično kartico ter geslo, ki ga pozna samo on. Četudi bi napadalcu uspelo ukrasti kartico, še vedno ne bi poznal gesla. Gesla s poskušanjem ne bi mogel pridobiti, saj se postopek avtentikacije po treh nepravilnih vpisih gesla onemogoči. Če pa bi napadalec poznal geslo in ne bi imel kartice, postopka avtentikacije sploh ne bi mogel začeti.

Cilj avtentikacije je torej potrditi identiteto odjemalca v realnem času. Pri komunikaciji med strežnikom in odjemalcem se je potrebno zavedati nevarnosti, da komunikacijo lahko prestreže srednji mož. V takem primeru govorimo o *napadu srednjega moža* (angl. man-in-the-middle attack). Srednji mož lahko prebere sporočila le v primeru, ko ta niso šifrirana. Takim situacijam se torej izognemo tako, da sporočila šifriramo. Da napadalcu še



Slika 2.1: Shema delovanja enostranskega avtentikacijskega protokola.

bolj otežimo delo, vsa sporočila v vsaki seji šifriramo z novim ključem, imenovanim *sejni ključ* (angl. session key). Izračunamo ga s pomočjo zasebnega ključa, ki ga poznata le strežnik in odjemalec.

Za avtentikacijo obstaja veliko različnih protokolov. Podrobneje si bomo ogledali protokol *avtentikacije izziv-odgovor* (angl. challenge/response authentication), pri katerem sodeluje tudi kartica SIM. Gre za protokol, kjer strežnik postavi *vprašanje* oziroma *izziv* (angl. challenge), odjemalec pa mora nanj podati pravilen *odgovor* (angl. response). Pri tem velja omeniti, da pri odgovoru ni nujno, da gre za skrivni podatek. Primer takega izziva je CAPTCHA, kjer se odjemalca sprašuje, kateri podatek je na sliki. Gre za enostaven primer Turingovega testa, s katerim določimo, ali je odjemalec človek ali stroj.

Strežnik najprej generira izziv S_c, ki ga pošlje odjemalcu. Odjemalec skupaj z zasebnim ključem K in znanim algoritmom h izračuna odgovor: $C_r = h(S_c, K)$, ki ga pošlje nazaj strežniku. Strežnik pozna zasebni ključ, zato lahko preveri, če je odjemalčev odgovor pravilen. Opisani protokol je primer enostranske avtentikacije.

Poglejmo še, kako bi morali dopolniti avtentikacijo, da bi postala dvostranska. Želimo, da tudi odjemalec preveri, če je strežnik pravi. Ponovno najprej strežnik generira izziv S_c, ki ga pošlje odjemalcu. Tudi odjemalec

	Strežnik		Odjemalec
1.	generira izziv S_c		
		pošlje izziv S_c $\longrightarrow$	
2.			generira izziv C_c in izračuna odgovor C_r s pomočjo zasebnega ključa $C_r = h(S_c, C_c, K)$
		$\longleftarrow$ pošlje odgovor C_r in izziv C_c	
3.	preveri pravilnost odgovora C_r in izračuna $S_r = h(S_c, C_c, K)$		
		pošlje odgovor S_r $\longrightarrow$	
4.			preveri pravilnost odgovora S_r

Slika 2.2: Shema delovanja dvostranskega avtentikacijskega protokola.

generira izziv C_c, ki ga uporabi pri izračunu odgovora: $C_r = h(S_c, C_c, K)$. Odjemalec strežniku pošlje vrednosti C_r in izziv C_c. Nato strežnik preveri, če je izračunani odgovor na izziv C_r pravilen, ter izračuna svoj odgovor $S_r = h(S_c, C_c, K)$, ki ga pošlje nazaj do odjemalca. Če tudi odjemalec potrdi ustreznost odgovora S_r je avtentikacija uspešno zaključena.

Avtentikacija izziv-odgovor rešuje vprašanje, kako izmenjevati sejne ključe. Izziv in zasebni ključ sta nujno potrebni komponenti za izračun sejnega ključa, ki ga uporabljamo v nadaljnji komunikaciji. S tem se zaščitimo pred omenjenim napadom srednjega moža. Kljub temu, da napadalec lahko s prisluškovanjem pridobi izziv, bo za izračun še vedno potreboval zasebni ključ, ki pa ga nima.

Poglavje 3

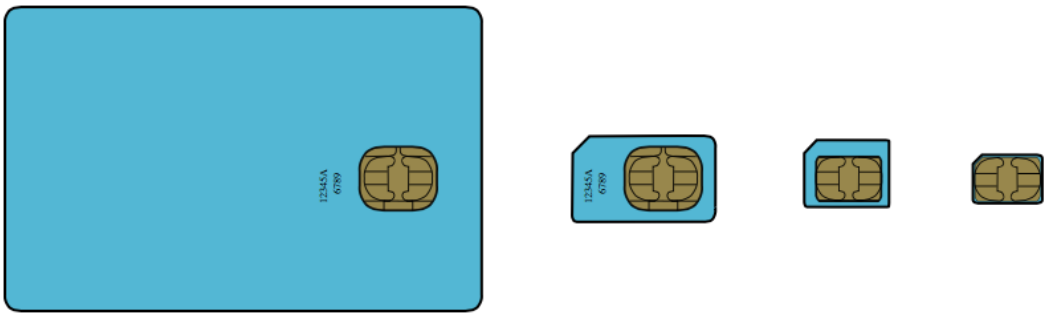
Kartica SIM

Kartica SIM (angl. subscriber identity module, SIM) je pametna kartica z vgrajenim mikroprocesorjem in pomnilnikom. Vstavimo jo v mobilni telefon, skupaj z njim pa sodeluje pri avtentikacijskem postopku uporabnika v mobilno omrežje. Prve kartice, v podobni velikosti kot kreditna kartica, so bile predstavljene leta 1991. Kasneje so se na trgu pojavile mini-, micro- in nano-SIM kartice. Vsaka se od prejšnje razlikuje po večjem pomnilniku in manjši velikosti, kar prikazuje slika 3.1. Kljub tehnološkemu razvoju je njeno delovanje še vedno zelo podobno kot na začetku.

V tem poglavju bomo predstavili kartico SIM. V razdelku 3.1 bomo opisali podatke, ki so shranjeni na njej. Sledil bo razdelek 3.2 s predstavitvijo protokola GSM, kjer bomo predstavili tudi vlogo kartice SIM. Poglavje je povzeto po [6, 15].

3.1 Podatki na kartici SIM

Mobilni operater nam skupaj s kartico SIM posreduje tudi številki PIN in PUK. Pri uporabi kartice SIM moramo paziti, da številke PIN trikrat ne vnesemo narobe, sicer se kartica zaklene. Odkleniti jo je mogoče s številko PUK, ki ponastavi števec nepravilnih vnosov številke PIN. Seveda pa se na kartici SIM skrivajo tudi številni drugi podatki.



Slika 3.1: Primerjava med formati kartice SIM. Vir: Wikipedia

3.1.1 Identifikacijska številka kartice SIM

Številka ICCID (angl. Integrated circuit card identifier) je do 22 mestna identifikacijska številka kartice SIM, ki je shranjena na kartici sami, prav tako pa je tudi odtisnjena na njej. Struktura številke ICCID je predstavljena v tabeli 3.1.

Dolžina	Primer	Opis
2 znaka	`89` `Telekomunikacije`	identifikacijska številka industrije, kjer se kartica SIM uporablja
1 - 3 znaki	`386` `Slovenija`	oznaka države
1 - 4 znaki	`40` `Simobil`	identifikacijska številka izdajatelja
do 12 znakov	`1234 1234 1234`	identifikacijska številka kartice
1 znak	`1`	kontrolna vsota

Tabela 3.1: Struktura identifikacijske številke kartice SIM.

Dolžina	Primer	Opis
3 znaki	`293` `Slovenija`	oznaka države
2-3 znaki	`40` `Simobil`	oznaka mobilnega omrežja
10 znakov	`1234567890`	identifikacijska številka mobilnega naročnika

Tabela 3.2: Struktura identifikacijske številke uporabnika.

3.1.2 Identifikacijska številka uporabnika

Številka IMSI (angl. international mobile subscriber identity) predstavlja identifikacijsko številko uporabnika v mobilnem omrežju. Struktura številke IMSI je prikazana v tabeli 3.2. Z njo se uporabnik predstavlja v mobilnem omrežju.

3.1.3 Zasebni ključ K_i

Zasebni ključ K_i je 128 bitna spremenljivka, shranjena na kartici SIM. Zasebnega ključa ni mogoče izpisati, uporabimo ga lahko le z algoritmom COMP128, ki skupaj z drugim vhodnim podatkom pripravi povzetek. Le-ta je uporaben pri avtentikaciji kartice SIM v mobilno omrežje. Prav tako se zasebni ključ K_i uporablja tudi pri generiranju sejnih ključev K_c, ki se uporabljajo za šifriranje podatkov, ki potujejo po mobilnem omrežju. Vsaka kartica SIM vsebuje unikaten zaseben ključ K_i, ki je shranjen na njej in v podatkovni bazi mobilnega operaterja.

3.1.4 Lokacija

Mobilno omrežje mobilnega operaterja je razdeljeno na več lokacijskih območij, od katerih ima vsako svojo številko (angl. location area identity, LAI). Njena

Dolžina	Primer	Opis
3 znaki	`293` `Slovenija`	oznaka države
2-3 znaki	`40` `Simobil`	oznaka mobilnega omrežja
16 bitno število	`CAAD35AAD73C8AAD` `985C92B24EEF1CFD`	oznaka bazne postaje (vsak operater ima lahko največ 65536 baznih postaj)

Tabela 3.3: Struktura identitete lokacijskega območja.

struktura je prikazana v tabeli 3.3. Kartica SIM shranjuje tudi podatke o omrežju, ki jih pridobi iz lokacijskega območja. Ko se mobilni telefon, v katerem je kartica SIM, premika, vsakič ko zamenja lokacijo, shrani podatke o novi lokaciji na kartico SIM, le-ta pa pošlje podatek o novi lokaciji nazaj v mobilno omrežje.

3.1.5 Stiki, kratka sporočila in drugi uporabniški podatki

Že od samega začetka kartica SIM omogoča tudi shranjevanje uporabniških podatkov. V začetku je kartica SIM shranjevala tudi zadnjih nekaj klicev, potem pa so se ti podatki začeli shranjevati v pomnilnik mobilnega telefona. Na kartico SIM je mogoče shraniti stike iz telefonskega imenika. Za razliko od mobilnega telefona si kartica SIM zapomni le osnovne parametre: telefonsko številko in ime osebe, ne pa tudi fotografije, elektronskega naslova, itd. Prednost shranjevanja stikov na kartici SIM pa je prenosljivost iz enega telefona na drugega. Ko kartico SIM premaknemo v drug telefon, s tem ne izgubimo stikov. Podobno je tudi s kratkimi sporočili. Žal pa je pomnilnik kartice SIM mnogo manjši od pomnilnika, ki ga premore mobilni telefon, zato se uporabniški podatki pogosto shranjujejo le v pomnilnik mobilnega telefona.

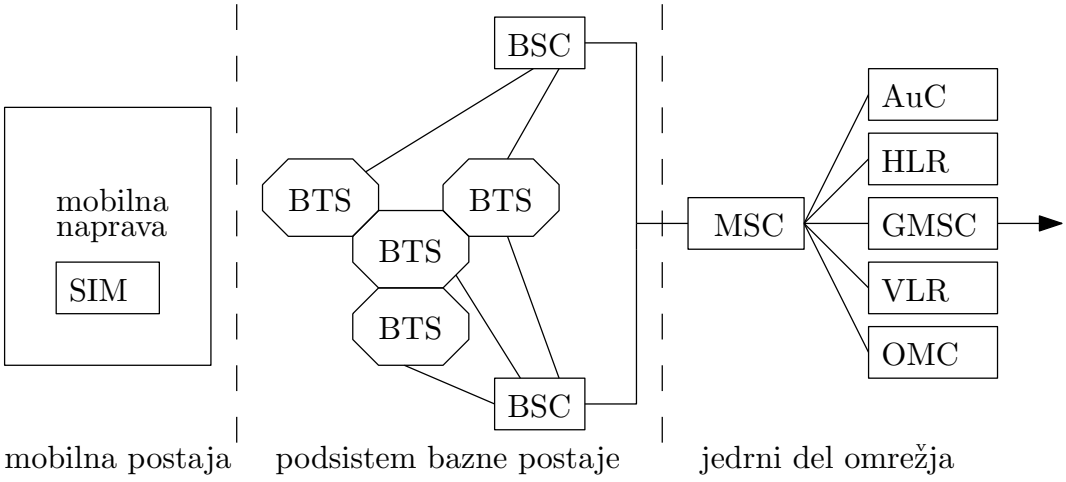
Poleg tega je mogoče na kartici SIM shranjevati tudi aplikacije mobilnega operaterja. Slednje imajo neposreden dostop do omrežja, njihovo izvajanje pa je neodvisno od operacijskega sistema telefona - le-ta na njih nikakor ne more vplivati. Popularnejše aplikacije ponudnikov omogočajo plačevanje, pri nas pa so bile pogosto v uporabi aplikacije, ki so omogočale preverjanje stanja na računu ter vzpostavitev povezave s klicnim centrom za pomoč uporabnikom. Posodabljanje aplikacij je mogoče na daljavo, brez vednosti uporabnika, zato le-te predstavljajo veliko varnostno luknjo.

3.2 Vloga kartice SIM pri protokolu GSM

V tem razdelku si bomo ogledali delovanje protokola GSM. Pojasnili bomo, kakšno vlogo ima kartica SIM v protokolu GSM. Sledili bomo knjigi [6].

Protokol nam pove, kakšna pravila in postopke moramo uporabiti v posameznih okoliščinah. *Protokol GSM* je torej obsežna zbirka pravil in postopkov, ki opredeljuje različne načine komunikacije, na primer komunikacijo med mobilnim telefonom in bazno postajo. Slika 3.2 nam prikazuje shemo protokola GSM. Mobilni telefon s kartico SIM se povezuje na posamezne *bazne sprejemno-oddajne postaje* (angl. base transceiving station). Njihovo delovanje nadzirajo *nadzorniki* (angl. base station controller), ki so povezani z *glavnimi preklopnimi centri* (angl. mobile switching center), ki se obnašajo kot klasično preklopno vozlišče analognih in digitalnih telefonskih central. Obravnavajo zahteve uporabnika, kot so klic, kratko sporočilo, in podobno.

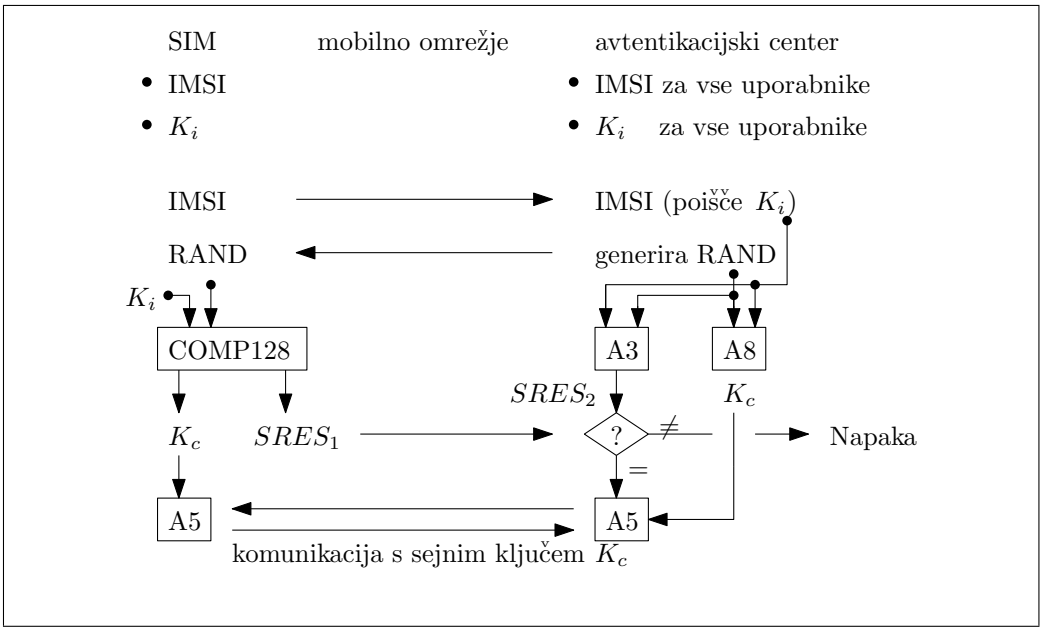
Po vklopu mobilne naprave je potrebna njena inicializacija, katere cilj je pripraviti mobilno napravo na komuniciranje z omrežjem. Inicializacijski postopek se začne s frekvenčno sinhronizacijo. Mobilna naprava išče radijske signale ter meri jakost njihovih signalov na frekvencah med 900 in 1800 MHz. Ko najde zadovoljiv signal za vzpostavitev zveze, preveri, če je to res iskani frekvenčni signal in nadaljuje s časovno sinhronizacijo. V tej fazi mobilna naprava pridobi osnovne podatke o povezani bazni postaji in mobilnem operaterju. Če mobilna naprava ugotovi, da se ne nahaja v domačem



Slika 3.2: Shema mobilnega omrežja.

omrežju, se bo postopek inicializacije znova začel od začetka. Ko bo mobilna naprava našla domače omrežje, sledi še zadnji korak, kjer pridobi informacije o omrežju in celici. Sedaj je mobilna naprava pripravljena na komunikacijo z omrežjem.

Ob prvi zahtevi mobilne naprave za klic ali drugo storitev, je potrebno izvesti avtentikacijo mobilne naprave v celoti. Po prvi uspešni avtentikaciji si mobilna naprava zapomni *šifrirno kodno sekvenčno številko* (angl. ciphering key sequence number), ki jo pozna tudi omrežje in postopek avtentikacije se preskoči. Mobilni telefon ob prvi zahtevi pošlje IMSI podatek iz kartice SIM do glavnega preklopnega centra, ki preveri, če je naprava že v *registru obiskovalcev* (angl. visitor location register) in obstaja 128 bitni naključni niz $RAND$, 32 bitni povzetek $SRES$ ter 64 bitni sejni ključ K_c. Če teh podatkov ni, jih zahteva v *domačem registru* (angl. home location register). Podatek IMSI omogoča iskanje zasebnega ključa K_i v registru. Glavni preklopni center nato preko mobilnega omrežja pošlje mobilni napravi naključni niz $RAND$ ter šifrirno kodno sekvenčno številko. Mobilna naprava z uporabo algoritma COMP128 na kartici SIM za naključni niz $RAND$ vrne povzetek $SRES_1$, ki ga posreduje nazaj do glavnega preklopnega centra. Tudi glavni preklopni center izračuna vrednost povzetka $SRES_2$ z enakima vhodnima



Slika 3.3: Shema avtentikacije v omrežje.

podatkoma $RAND$ ter K_i. Ključ K_i je shranjen v zavarovani bazi podatkov, ki jo upravlja *center za avtentikacijo* (angl. authentication center). Če sta povzetka $SRES_1$ in $SRES_2$ enaka, potem je mobilna naprava uspešno opravila avtentikacijo v mobilno omrežje. Pri naslednji zahtevi bo glavni preklopni center v registru obiskovalcev našel podatke o kartici SIM, zato ji bo posredoval samo šifrirno kodno sekvenčno številko ter preskočil avtentikacijo. V postopku avtentikacije kartice SIM se zgenerira tudi 64 bitni sejni ključ Kc, ki se ga uporablja kot vhod za družino algoritmov A5, ki skrbijo za šifriranje nadaljnje komunikacije. Veljavnost sejnega ključa je odvisna od nastavitev glavnega preklopnega centra in registra obiskovalcev posameznega mobilnega operaterja. Če bi se postopek avtentikacije izvedel pred vsako zahtevo, bi za vsak klic ali drugo storitev vedno uporabili drug sejni ključ. V praksi pa je pogosto isti sejni ključ uporabljen večkrat. Shema avtentikacije je prikazana na sliki 3.3.

Prekomernemu pošiljanju številke IMSI pa se izognemo z uvedbo številke TMSI (angl. temporary mobile subscriber identity), kar otežuje prisluškovanje

pogovorom v omrežju. Številka TMSI je začasna številka IMSI, ki jo generira glavni preklopni center po opravljeni avtentikaciji in jo shrani v register obiskovalcev ter pošlje kartici SIM. Dodaljena številka TMSI se uporablja pri nadaljnjih identifikacijah namesto številke IMSI.

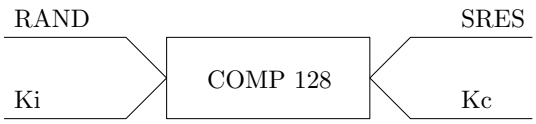
Ko se z mobilnim telefonom premikamo, menjamo tudi celice mobilnega omrežja. Mobilni telefon ves čas preverja, ali se podatek o lokacijskem območju omrežja ujema s podatkom o lokacijskem območju, ki je shranjen na kartici SIM. Če mobilni telefon zazna spremembo, v omrežje ponovno pošlje sporočilo z novo lokacijo. Glavni preklopni center sporočilo prejme ter popravi podatke v registru obiskovalcev. Redno osveževanje lokacijskih informacij v registru obiskovalcev nam omogoča spremljanje, ali je mobilni telefon še priključen v omrežje. V kolikor mobilni telefon preneha z osveževanjem lokacijskih informacij, glavni preklopni center predvideva, da je mobilni telefon izključen ter posodobi stanje v registru obiskovalcev.

Poglavje 4

Algoritem COMP128

Algoritem COMP128 je kombinacija algoritmov A3 in A8, ki sta definirana v protokolu GSM. Algoritem A3 pripravi povzetek v postopku avtentikacije, algoritem A8 pa generiran sejni ključ, ki se ga nato uporabi kot vhod za družino algoritmov A5, ki skrbijo za šifriranje komunikacij. Pri njegovem razvoju ni bilo upoštevano Kerchoffsovo načelo, saj njegova uradna specifikacija ni bila javno objavljena. Algoritem so na spletu objavili M. Briceno, I. Goldberg in D. Wagner, ki so ga pridobili s pomočjo inverznega inženiringa [2]. Kmalu zatem je bil na njegovo zgradbo izveden tudi napad, ki priča o slabih varnostnih lastnostih algoritma [1].

Od začetka uporabe do danes je algoritem doživel nekaj izboljšav, zato ga poznamo v štirih različicah. Prva različica (COMP128-1) ima slabo zgradbo, saj ni dosežena dovolj dobra razpršenost. Poleg tega je dejanska dolžina sejnega ključa K_c, ki ga dobimo na izhodu, le 54 bitov. Zadnjih 10 bitov je vedno praznih, kar predstavlja varnostno pomanjkljivost pri nadaljnji uporabi za šifriranje, saj je 54 bitov v današnjem času premalo in lahko ključ poiščemo s pregledovanjem vseh ključev. Sledila mu je druga različica (COMP128-2) s kompleksnejšim algoritmom, ki pa ni odpravila varnostne pomanjkljivosti z dolžino ključa K_c. To pomanjkljivost je odpravila šele tretja različica (COMP128-3). Obstaja tudi različica COMP128-4, ki uporablja AES. V nadaljevanju bomo opisali prvo različico algoritma COMP128 in napad nanj.



Slika 4.1: Shema delovanja algoritma COMP128.

4.1 Predstavitev algoritma

Algoritem COMP128 je kompresijska funkcija, ki za vhodna podatka potrebuje 128 bitni naključni niz, imenovan $RAND$, ki ga dobimo kot izziv iz mobilnega omrežja, ter 128 bitni zasebni ključ Ki, ki je shranjen na kartici SIM. Na izhodu algoritem vrne dve vrednosti: 32 bitno vrednost $SRES$, ki predstavlja povzetek za $RAND$, ter 64 bitni sejni ključ Kc, ki ga mobilni telefon v nadaljevanju uporabi za šifriranje pri algoritmu A5. Delovanje algoritma prikazuje slika 4.1.

Algoritem se izvede v osmih skoraj enakih krogih. Na začetku vhodne podatke shranimo v spremenljivko x. Med izvajanjem algoritma jih sproti prepisujemo z novo izračunanimi vrednostmi. Gre za 32 bajtni seznam, kjer je vsak element seznama predstavljen z 8 bitnim številom. Potrebujemo tudi spremenljivko b, v katero shranjujemo bitno predstavitev spremenljivke x. Algoritem 2 nam prikazuje psevdo kodo.

Vsak krog je sestavljen iz treh korakov: kompresija, pretvorba iz bajtov v bite ter permutacija. Slednje v zadnem krogu ne izvedemo. V vsakem koraku najprej ponastavimo prvo polovico seznama x z vrednostmi zasebnega ključa K_i. Sledi postopek kompresije, ki ga ponovimo petkrat. Rezultat kompresijske funkcije je potrebno pretvoriti iz bajtov nazaj v bitni format. Pri tem bitne vrednosti nato uporabimo pri permutaciji. Na koncu imamo v spremenljivki x shranjeno 128 bitno vrednost. Prvih 32 bitov predstavlja vrednost $SRES$, ki jo vrnemo kot povzetek naključne vrednosti $RAND$. Zadnjim 54 bitom dodamo še 10 praznih bitov ter vse skupaj vrnemo kot sejni ključ K_c.

vhod : Naključni 128 bitni niz $RAND$ in 128 bitni zasebni ključ K_i
izhod: 32 bitni povzetek $SRES$ in 64 bitni sejni ključ K_c

```
x [16...31] = RAND;
for i ← 0 to 8 do
    x [0...15] = Ki;
    x = Kompresija (x);
    b = Pretvorba (x);
    if i < 7 then
        Permutacija (x, b);
    end
end
PripraviIzhod (x, b);
```

Algoritem 2: COMP128.

Razložimo delovanje kompresijske in permutacijske funkcije podrobneje:

- **Kompresija**

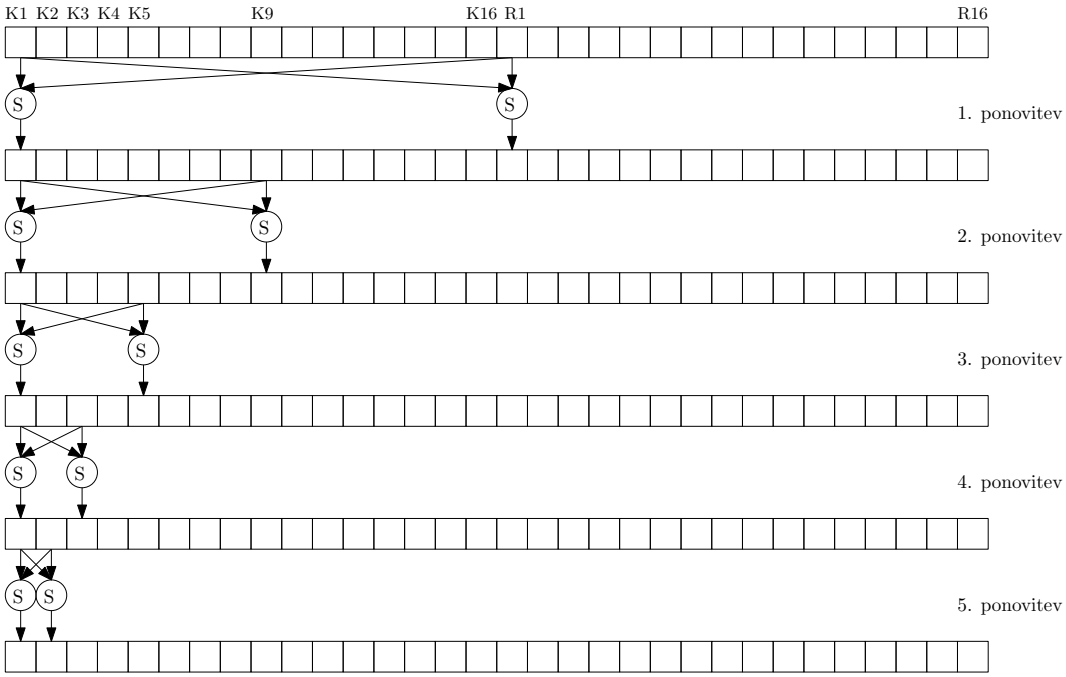
  Vhodni podatek za kompresijsko funkcijo je 32 bajtni seznam x. Kompresijska funkcija se izvede petkrat in nam na izhodu vrne 128 bitno število. Pred prvo iteracijo razdelimo seznam x na dva odseka. Pri vsaki nadaljnji iteraciji k posamezen odsek razpolovimo. Tako v drugi iteraciji dobimo štiri odseke, v tretji osem, v četrti šestnajst ter nazadnje dvaintrideset.

  V vsaki iteraciji izvedemo tudi operacijo S, ki jo prikazuje slika 4.2: glede na lego združimo elementa a in b iz sosednjih odsekov po naslednjem pravilu:

  $$l = a + 2 \cdot b \mod 2^{9-k}, \tag{4.1}$$

  $$r = 2 \cdot a + b \mod 2^{9-k}, \tag{4.2}$$

  kjer l predstavlja združitev v prvem, levem odseku ter r združitev v drugem, desnem odseku. Izračunani vrednosti l in r predstavljata indeksa elementov iz substitucijske tabele T_k (Dodatek A). Uporaba



Slika 4.2: Shema delovanja kompresijske funkcije.

substitucijske tabele nam zagotavlja, da zveza med biti ni linearna, za razpršenost pa poskrbita pravili (4.1) in (4.2). Vrednost modula v posamezni ponovitvi je merilo za število elementov, ki jih vsebuje posamezna substitucijska tabela T_k, in velikost elementov spremenljivke x. Z vsako iteracijo se velikost posameznega elementa v seznamu x zmanjšuje. Po prvi iteraciji je posamezen element predstavljen z osmimi biti, nato pa z vsako iteracijo izgubi en bit. Na koncu je posamezen element predstavljen s štirimi biti. Vse to lahko razberemo tudi iz algoritma 3.

Z vrednostima, ki ju pridobimo iz substitucijske tabele, prepišemo prvotni vrednosti na mestih a in b. S tem se izvajanje kompresijske funkcije zaključi. Shema delovanja algoritma je prikazana na sliki 4.2, kjer je vidna *oblika metulja* (angl. butterfly structure).

```
for i ← 0 to 4 do
    for j ← 0 to 2^i − 1 do
        for k ← 0 to 2^(4−i) − 1 do
            a = k + j · 2^(5−i);
            b = a +2^(4−i);
            l = (x[a] + 2 · x[b]) mod 2^(9−i);
            r = (2 · x[a] + x[b]) mod 2^(9−i);
            x[a] = Ti[l];
            x[b] = Ti[r];
        end
    end
end
```

Algoritem 3: Kompresija.

- **Permutacija**

  Permutacijska funkcija kot vhodna podatka vzame seznam x ter njegovo predstavitev v bitni obliki b. Funkcija posameznim elementom druge polovice seznama x spremeni vrednosti nekaterih bitov. Nove vrednosti se nahajajo v bitni predstavitvi b ter jih za ustrezni element x izberemo po pravilu:

  $$x[j+16][k] = b[((8 \cdot j + k) \cdot 17) \mod 128],$$

  kjer k zavzame vrednosti med 0 in 7.

  Funkcija nam vrne permutiran x, kar poveča razpršenost med biti. Njeno delovanje je prikazano z algoritmom 4.

```
for j ← 0 to 15 do
    for k ← 0 to 7 do
        x [j + 16][k] = b[((8 · j + k) · 17) mod 128)];
    end
end
```

Algoritem 4: Permutacija.

Primer poteka algoritma je prikazan na sliki 4.3. Na začetku sta podana zasebni ključ K_i ter naključni niz $RAND$, ki vstopita v algoritem COMP128. Izpisana sled algoritma prikazuje stanje spremenljivke x po končanih kompresijah. Vrednosti so predstavljene v šestnajstiškem zapisu. Sledi vrednost spremenljivke b po končani pretvorbi iz bajtov v bite v dvojiškem zapisu. Vrednost funkcije x po permutaciji je znova prikazana v šestnajstiškem zapisu. Potek v naslednjih sedmih iteracijah je podoben. Le v zadnji iteraciji se ne izvede permutacija na spremenljivki x. Na izhodu tako dobimo povzetek $SRES$ ter sejni ključ K_c, ki sta v zgledu predstavljena v šestnajstiškem zapisu.

4.2 Napad z grobo silo in rekonstrukcija zasebnega ključa K_i

V tem razdelku bomo opisali napad na algoritem COMP128, ki izkoristi premajhno razpršenost pri kompresijski funkciji. Omogoča nam rekonstrukcijo zasebnega ključa K_i. To je zelo nevarno, saj z njim lahko naredimo dvojnik kartice SIM ter opravljamo klice v imenu dejanskega lastnika kartice SIM. Posedovanje zasebnega ključa nam omogoča dešifriranje podatkov, ki smo jih zajeli med predhodnim prestrezanjem podatkov v mobilnem omrežju.

Za izvedbo napada na algoritem COMP128 potrebujemo fizični dostop do kartice SIM.

Napad bomo izvedli z grobo silo. Pametno bomo izbirali izzive, kar nam bo omogočalo, da bomo lahko korak za korakom rekonstruirali celoten zasebni ključ K_i, ki ga napadalec ne pozna, ki je shranjen na kartici SIM. Ker imamo dostop do kartice SIM, si lahko naključni niz $RAND$ izmislimo sami ter ga posredujemo kodirnemu sistemu (naši kartici SIM) ter spremljamo rezultate. Za dva različna vhoda $RAND_1$ in $RAND_2$ bomo poskušali najti enak rezultat:

$$COMP128(RAND_1) = COMP128(RAND_2).$$

Vhodni podatki

Skrivni ključ K_i:	21 59 E7 41 95 F3 51 5A DC 1A D0 64 24 F2 C9 FC
RAND:	D8 F7 F2 02 E9 61 DA 0C 6E 61 06 7F F3 0A BE 7B

1. iteracija

Kompresija #1:	39 0E 34 01 E8 7B 9C 65 EA 7F 7F 27 08 4B 01 6F
	DF D9 68 1D 35 0E 16 68 06 2D E5 31 A6 17 B6 28
Kompresija #2:	03 61 1C 4E 21 7C 07 60 5C 28 3D 7E 06 50 44 44
	1D 2D 1C 4B 3F 6E 1C 45 41 7E 3A 6E 1B 2D 0D 21
Kompresija #3:	2A 2E 3E 34 03 10 38 3F 38 22 2A 29 10 37 10 30
	0F 33 37 2E 35 22 37 18 05 36 37 3E 16 0D 32 1D
Kompresija #4:	1C 03 18 1C 1E 06 16 09 18 1A 0E 16 11 0D 11 08
	05 1F 1B 04 1A 18 00 04 1E 0C 05 0F 0D 17 08 1F
Kompresija #5:	0A 09 0A 01 0E 0A 05 00 01 0E 08 04 07 08 0C 0E
	04 00 04 08 0E 01 05 01 03 05 04 06 09 03 0B 08
Pretvorba iz bajtov v bite:	1010 1001 1010 0001 1110 1010 0101 0000
	0001 1110 1000 0100 0111 1000 1100 1110
	0100 0000 0100 1000 1110 0001 0101 0001
	0011 0101 0100 0110 1001 0011 1011 1000
Permutacija:	0A 09 0A 01 0E 0A 05 00 01 0E 08 04 07 08 0C 0E
	D1 CA A7 05 40 D0 72 C8 7A 50 8C 66 29 35 B9 88

⋮

Izhodni podatki

SRES:	76 6F B1 FB
Sejni ključ K_c:	D8 3A A3 A4 7C 76 78 00

Slika 4.3: Potek algoritma COMP128.

Vhodni podatki

RAND1: 11 00 00 00 00 00 00 00 11 00 00 00 00 00 00 00

RAND2: FF 00 00 00 00 00 00 00 FF 00 00 00 00 00 00 00

1. iteracija

Kompresija #1:

RAND1: 54802CC22D6FF311FDA4395740865EC35CA805CB712C267BEC3FA5EC5F560082

RAND2: A1802CC22D6FF31141A4395740865EC3F7A805CB712C267B913FA5EC5F560082

Kompresija #2:

RAND1: 133E070C36084A10660C6B257516176A63244E700606244B0C1D4A1C50376526

RAND2: 443E070C36084A10310C6B257516176A59244E700606244B4B1D4A1C50376526

Kompresija #3:

RAND1: 07110F071E1536391D291735022C13081B3E1C2918111300E07203D38383535 20

RAND2: 3E110F07101536390F291735112C1308233E1C2929111300E30203D38013535 20

⋮

Slika 4.4: Primerjava kompresij prve iteracije za različna vhoda.

Iščemo torej trk. Trk pri kompresijskih funkcijah je mogoče najti, ker kompresijska funkcija iz večje množice slika v manjšo (Dirichletovo načelo). Pri tem bomo upoštevali tudi paradoks rojstnega dne, ki smo ga že opisali. Paradoks rojstnega dne pravi, da ob n bitni dolžini povzetka, kar pomeni 2^n možnih različnih povzetkov, z verjetnostjo večjo od 0.5 pride do trka ob približno $2^{n/2}$ naključnih nizih. Za 128 bitne povzetke torej velja, da se bo med 2^{64} povzetki naključnih nizov pojavil trk z verjetnostjo večjo od 0.5.

Pri kompresijski funkciji bomo izkoristili še strukturo metulja. Kompresijska funkcija v vsaki ponovitvi 32 bajtno vhodno spremenljivko x z vsako ponovitvijo razpolovi na odseke. Najprej jo razdelimo na dva odseka, ki sta velika 16 bajtov. V naslednji ponovitvi imamo štiri odseke velikosti 8 bajtov.

Vhodni podatki	
RAND1:	3E 00 00 00 00 00 00 00 C1 00 00 00 00 00 00 00
RAND2:	1D 00 00 00 00 00 00 00 99 00 00 00 00 00 00 00
Izhodni podatki	
SRES in K_c za RAND1:	80 B3 A7 6A D1 21 F6 69 03 D0 F8 00
SRES in K_c za RAND2:	80 B3 A7 6A D1 21 F6 69 03 D0 F8 00

Slika 4.5: Trk pri vhodnih parametrih $RAND_1$ in $RAND_2$.

Nato dobimo osem odsekov po 4 bajte ter z novo ponovitvijo že 16 odsekov po 2 bajta. Na koncu imamo 32 odsekov velikosti 1 bajta. V vsakem krogu se podatka iz dveh sosednjih odsekov uporabita za izračun indeksa, ki nam pove, katero vrednost iz substitucijskih tabel T uporabimo in jo zapišemo nazaj v spremenljivko x namesto prvotne vrednosti. Končni rezultat izračuna indeksa je vedno odvisen le od dveh vhodnih podatkov.

Slika 4.4 nam prikazuje začetek sledi izvajanja algoritma za dve različni vhodni vrednosti $RAND_1$ in $RAND_2$. Podrobneje si poglejmo, kakšno je stanje po koncu druge ponovitve. V drugi ponovitvi so mesta $i, i+8, i+16, i+24$ odvisna samo od vrednosti na teh mestih. Notranje stanje je torej odvisno le od štirih bajtov $i, i+8, i+16, i+24$, zato ne dosežemo zadostne razpršenosti. Le-to bi lahko dosegli, če bi uporabili vse bajte. Vemo, da v algoritem vstopi spremenljivka x, velikosti 32 bajtov, ki je sestavljena iz zasebnega ključa K_i ter naključnega niza $RAND$. Tako mesti $i, i+8$ pripadata prvemu ter osmemu bajtu zasebnega ključa, mesti $i+16, i+24$ pa prvemu ter osmemu bajtu naključnega niza.

Iskanja trka se bomo lotili z algoritmom 1, ki smo ga opisali v razdelku 2.2.2. Naključno izbrano vrednost, ki ji bomo spreminjali vrednosti na mestih $i, i+8$, bomo poslali kot izziv funkciji COMP128. Ker so vse operacije znotraj kompresijske funkcije odvisne le od štirih bajtov, se bo trk ohranjal skozi vse ponovitve. Če se bo torej trk pojavil po drugi ponovitvi, ga bomo zaznali tudi

v povzetku. Pri napadu bomo v zgoščeni tabeli shranjevali izzive, dokler ne bomo našli dveh različnih izzivov z enakim povzetkom. Primer dveh različnih izzivov z enakim povzetkom je predstavljen na sliki 4.5.

Ko najdemo vrednosti $RAND_1$ in $RAND_2$, fiksiramo mesti $i + 16$ in $i + 24$ ter začnemo z iskanjem mest i ter $i + 8$ zasebnega ključa K_i. Ključe K_i preiskujemo po vrsti z menjavanjem vrednosti na mestih i in $i+8$, dokler ne velja:

$$COMP128(RAND_1, K_i) = COMP128(RAND_2, K_i).$$

Postopek, ki ga prikazuje algoritem 5, moramo ponoviti osemkrat, saj v vsaki ponovitvi iščemo vrednosti na dveh mestih $i, i + 8$. Ker iščemo trk po koncu druge ponovitve, je vsak element spremenljivke x velik 7 bitov. Z upoštevanjem paradoksa rojstnega dne bomo trk našli v približno $2^{\frac{4 \cdot 7}{2}} = 2^{14}$ iskanjih. Zaradi osemkratne ponovitve potrebujemo $8 \cdot 2^{14} = 2^{17}$ iskanj.

Za napad v praksi potrebujemo čitalec za kartico SIM, ki nam omogoča, da kartici SIM pošiljamo izzive in nam vrača njihove povzetke. Kartica SIM lahko v povprečju v eni sekundi odgovori na šest izzivov.

Testno simulacijo napada smo opravili na MacBook Pro s procesorjem Intel Core i7 2,3 GHz. Algoritem COMP128 in algoritma za iskanje trkov ter rekonstrukcijo ključa smo implementirali v programskem jeziku Python 2.7. V eni sekundi lahko izračunamo približno 100 povzetkov, kar pomeni, da za posamezno mesto potrebujemo približno tri minute. Zaradi omejitve kartice SIM, ki v eni sekundi izračuna le šest povzetkov, postopek traja veliko dlje časa, približno eno uro. Za izračun osmih parov trkov na računalniku potrebujemo manj kot pol ure, pri uporabi kartice SIM pa približno šest ur.

Za izračunane povzetke pa je potrebno rekonstruirati tudi zasebni ključ K_i. Proces rekonstrukcije v vsakem primeru izvedemo na računalniku. Preveriti moramo $8 \cdot 2^{16} = 2^{19}$ različnih ključev, v eni sekundi pa lahko preverimo šestdeset ključev. Rekonstrukcija se konča v slabih treh urah. Ker gre za napad z grobo silo, kjer po vrsti pregledujemo vse možnosti, je trajanje izvajanja močno odvisno od vrednosti na posameznem mestu i. Trajanje postopka pri manjših vrednostih tako močno skrajša čas iskanja.

```
vhod : ponovitev i, RAND1 in RAND2
izhod: Kc
for j ← 0 to 2^8 do
    for k ← 0 to 2^8 do
        Kc[i] = dec2hex(j);
        Kc[i + 8] = dec2hex(k);
        if COMP128(RAND1, Kc) = COMP128(RAND2, Kc) then
            return Kc
        end
    end
end
```

Algoritem 5: Iskanje zasebnega ključa z grobo silo.

Ko najdemo osem parov trkov ter njim pripadajoče odseke ključa, poznamo zasebni ključ K_i. To nam omogoča, da lahko naredimo kopijo kartice SIM in se z njo avtenticiramo v omrežje. S tem je naše delo, kot napadalec, zaključeno. Sedaj lahko začnemo s prisluškovanjem klicem, klicanjem v imenu lastnika kartice SIM, pošiljanjem kratkih sporočil in podobno.

Po odkritju napada so izdelovalci v kartico SIM vgradili mehanizem, ki preprečuje opisan napad. Mehanizem po 2^{16} poizkusih zaklene kartico SIM, tako da je ni več mogoče uporabljati. Zaradi slabih lastnosti algoritma COMP128-1 sta ga kmalu zamenjala algoritma COMP128-2 in COMP128-3. Algoritem COMP128-1 se danes ne uporablja več.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo na kratko predstavili delovanje mobilnega omrežja in vlogo kartice SIM pri avtentikaciji v omrežje. Podrobno smo opisali algoritem COMP128, ki izračuna povzetke, ki se jih uporabi pri avtentikaciji, ter sejni ključ. Čeprav naj bi avtentikacija in šifriranje podatkov zagotavljalo varno komunikacijo, se izkaže, da temu ni tako.

Eno od pomanjkljivosti predstavlja tudi slaba struktura algoritma COMP-128, kar smo pokazali s preprostim napadom, ki deluje v realnem času in omogoča rekonstrukcijo zasebnega ključa. Napad smo izvedli na osebnem računalniku. Uporabili smo programski jezik Python, v katerem smo najprej implementirali algoritem COMP128, nato pa še algoritma za napad in rekonstrukcijo zasebnega ključa. Postopek rekonstrukcije zasebnega ključa bi lahko še pohitrili z uporabo vzporednih procesov.

Algoritem COMP128 je doživel nekaj nadgradenj in izboljšav, prav tako so proizvajalci kartic SIM poskrbeli za dodatno zaščito, ki preprečuje podvajanje kartic. Kljub temu pa mobilno omrežje omogoča, da se avtentikaciji uporabnika s kartico SIM popolnoma izognemo in ponaredimo identiteto uporabnika.

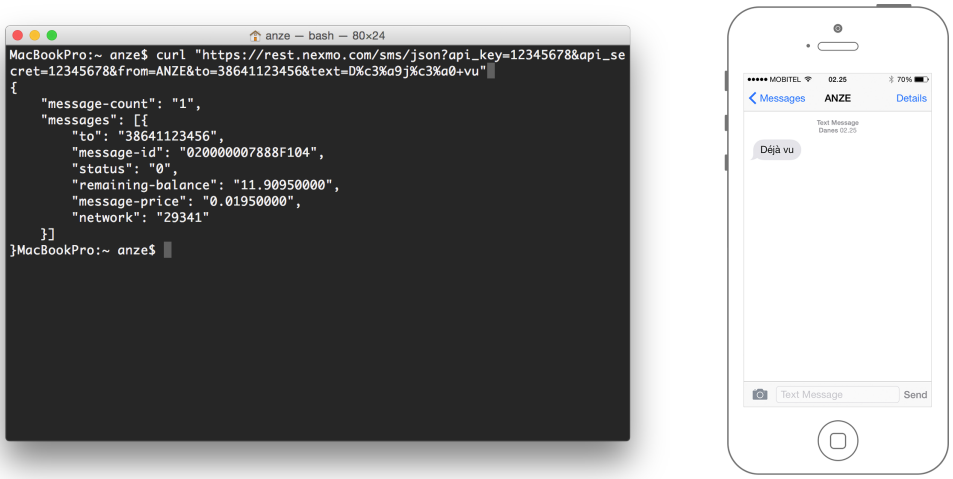
Prve prevare s ponarejeno identiteto pošiljatelja (angl. caller ID) so se začele dogajati takoj po uvedbi digitalne telefonije. Sprva sicer ni šlo za prevare, pač pa so ga uporabljala podjetja, ki so imela dostop do dragega PRI

(angl. primary rate interface). Le-ta jim je omogočal dostop do več različnih telefonskih linij, ki so jim lahko ob izhodnih klicih nastavili, katero številko bo videl klicoči. Storitev so uporabljala z namenom, da ob neodgovorjenem klicu klicoči pokliče na telefonsko centralo, od koder ga preusmerijo do ustrezne osebe.

Proti koncu 20. in v začetku 21. stoletja so se nad storitvijo navdušili zasebni preiskovalci, ki so zakupili več telefonskih linij preko PRI ter jih nato preprodajali kot slepe linije (angl. blind lines). Njihovi naročniki so se storitev posluževali, ker se pri klicu strankam ni pokazala njihova dejanska telefonska številka in jim je to zagotavljalo zasebnost.

Kasneje so nekateri ponudniki telefonije uporabnikom onemogočili spreminjanje identitete pošiljatelja na način, da so ne glede na uporabnikove zahteve klicu dodali podatek iz svoje podatkovne baze. Z razmahom ponudnikov VoIP (angl. Voice over IP) je možnost ponarejanja identitet znova postala aktualna. Ponudniki VoIP sicer ponujajo zamenjavo identitete pošiljatelja z namenom, da klicoči vedno vidi na primer številko našega mobilnega telefona, četudi klic opravimo preko računalnika. Pred lažnimi identitetami so se zaščitili tako, da je pred pošiljanjem poljubne identitete potrebno opraviti njeno preverjanje. V kolikor je uspešno, potem identiteto lahko uporabljajo, če pa so izbrali lažno telefonsko številko, preverjanje ni uspešno in identitete ne morejo uporabiti.

Kljub temu pa na trgu obstajajo ponudniki, ki omogočajo spremembo identite pošiljatelja brez ustreznega preverjanja. Gre za podjetja, ki ponujajo inovativne komunikacijske vmesnike, ki povezujejo tradicionalni govor in kratka sporočila z oblačnimi komunikacijami. Za dostop do njihovega vmesnika je največkrat potrebna zgolj brezplačna registracija, kjer pustimo svoje ime, priimek ter elektronski naslov. Po uspešni zaključeni registraciji dobimo podatke za dostop do vmesnika. Z uporabo vmesnika lahko hitro in enostavno pošljemo na primer kratko sporočilo, kjer za pošiljatelja navedemo lažno ime. Takšen primer nam prikazuje slika 5.1. V kolikor za pošiljatelja navedemo telefonsko številko v mednarodnem formatu, jo bo mobilni tele-



Slika 5.1: Levo: prikaz odgovora po uspešni uporabi vmesnika, ki omogoča pošiljanje kratkega sporočila. Desno: prejeto sporočilo na mobilnem telefonu.

fon, ki si ga lasti žrtev, poiskal v imeniku, ji dodal ime stika ter omogočal, da žrtev stik tudi pokliče.

Žal se povprečen uporabnik premalo zaveda nevarnosti, ki prežijo nanj ob uporabi mobilne telefonije. V letu 2012 je bila izvedena varnostna analiza slovenskih mobilnih omrežij, ki je pokazala, da so mobilni operaterji še vedno uporabljali slabe šifrirne algoritme [5], ki so omogočali precej enostavno prisluškovanje. Glede na željo po vse večji zasebnosti, bo vsekakor zanimivo spremljati spremembe na področju varnosti mobilne telefonije v naslednjih letih. Upajmo, da bodo novi varnostni protokoli odpravili pomanjkljivosti starih in omogočali varno uporabo vsem uporabnikom.

Literatura

[1] M. Briceno, I. Goldberg and D. Wagner, "GSM Cloning", *Web page about COMP-128 version 1*, 1998. `http://www.isaac.cs.berkeley.edu/isaac/gsm.html`. [Dostopano 11. 9. 2015].

[2] M. Briceno, I. Goldberg and D. Wagner, "Implementation of COMP128", 1998. `http://www.scard.org/gsm/a3a8.txt`. [Dostopano 11. 9. 2015].

[3] B. Brumley, "A3/A8 & COMP128", *T-79.514 Special Course on Cryptology*, 2004. `http://www.tcs.hut.fi/Studies/T-79.514/slides/S5.Brumley-comp128.pdf`. [Dostopano 11. 9. 2015].

[4] H. Handschuh and P. Paillier, "Reducing the collision probability of alleged Comp128", *Lecture Notes in Computer Science*, zv. 1820, str. 380–385, 2000.

[5] J. Hudolin, M. Kovačič, K. Rupnik, "Varnost slovenskih GSM omrežij", 2012. `https://slo-tech.com/clanki/12003/`. [Dostopano 11. 9. 2015].

[6] G. Heine, *GSM networks: Protocols, terminology and implementation*, Artech House, Inc., 1999.

[7] A. Kerckhoffs, "La cryptographie militaire", *Journal des sciences militaires*, zv. IX, str. 5–83, januar 1883.

[8] A. Kerckhoffs, "La cryptographie militaire", *Journal des sciences militaires*, zv. IX, str. 161–191, februar 1883.

[9] A. J. Menezes, P. C. Van Oorschot and S. A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1996.

[10] M. Y. Rhee, *Mobile Communication Systems and Security*, John Wiley & Sons, 2009, str. 10–16.

[11] D. R. Stinson, *Cryptography: Theory and practice*, 3. izdaja, Chapman & Hall/CRC, 2006.

[12] S. Wray, "COMP128: A birthday surprise", 2003. `http://www.stuartwray.net/comp128-a-birthday-surprise-rev.pdf`. [Dostopano 11. 9. 2015].

[13] Y. Zhou, Y. Yu, F.-X. Standaert and J.-J. Quisquater, "On the Need of Physical Security for Small Embedded Devices: A Case Study with COMP128-1 Implementations in SIM Cards (long version)", 2013. `https://eprint.iacr.org/2013/224.pdf`. [Dostopano 11. 9. 2015].

[14] Wikipedia. COMP128. `https://en.wikipedia.org/wiki/COMP128`. [Dostopano 11. 9. 2015].

[15] Wikipedia. Subscriber identity module. `https://en.wikipedia.org/wiki/Subscriber_identity_module`. [Dostopano 11. 9. 2015].

Dodatek A

Substitucijske tabele

Kompresijska funkcija algoritma COMP128 uporablja substitucijske tabele. Gre za tabelarično predstavitev funkcije f_i, ki slika iz indeksne množice v množico vrednosti:

$$f_i : \{0, 1, \ldots, 2^{9-i}\} \rightarrow \{0, 1, \ldots, 2^{8-i}\}.$$

Vrednosti funkcije f_i so zastopane z enako verjetnostjo in so predstavljene v tabeli T_i.

Tabela A.1: Substitucijska tabela T_0

102	177	186	162	2	156	112	75	55	25	8	12	251
193	246	188	109	213	151	53	42	79	191	115	233	242
164	223	209	148	108	161	252	37	244	47	64	211	6
237	185	160	139	113	76	138	59	70	67	26	13	157
63	179	221	30	214	36	166	69	152	124	207	116	247
194	41	84	71	1	49	14	95	35	169	21	96	78
215	225	182	243	28	92	201	118	4	74	248	128	17
11	146	132	245	48	149	90	120	39	87	230	106	232
175	19	126	190	202	141	137	176	250	27	101	40	219
227	58	20	51	178	98	216	140	22	32	121	61	103
203	72	29	110	85	212	180	204	150	183	15	66	172
196	56	197	158	0	100	45	153	7	144	222	163	167

Tabela A.2: Substitucijska tabela T_0, nadaljevanje

60	135	210	231	174	165	38	249	224	34	220	229	217
208	241	68	206	189	125	255	239	54	168	89	123	122
73	145	117	234	143	99	129	200	192	82	104	170	136
235	93	81	205	173	236	94	105	52	46	228	198	5
57	254	97	155	142	133	199	171	187	50	65	181	127
107	147	226	184	218	131	33	77	86	31	44	88	62
238	18	24	43	154	23	80	159	134	111	9	114	3
91	16	130	83	10	195	240	253	119	177	102	162	186
156	2	75	112	25	55	12	8	193	251	188	246	213
109	53	151	79	42	115	191	242	233	223	164	148	209
161	108	37	252	47	244	211	64	237	6	160	185	113
139	138	76	70	59	26	67	157	13	179	63	30	221
36	214	69	166	124	152	116	207	194	247	84	41	1
71	14	49	35	95	21	169	78	96	225	215	243	182
92	28	118	201	74	4	128	248	11	17	132	146	48
245	90	149	39	120	230	87	232	106	19	175	190	126
141	202	176	137	27	250	40	101	227	219	20	58	178
51	216	98	22	140	121	32	103	61	72	203	110	29
212	85	204	180	183	150	66	15	196	172	197	56	0
158	45	100	7	153	222	144	167	163	135	60	231	210
165	174	249	38	34	224	229	220	208	217	68	241	189
206	255	125	54	239	89	168	122	123	145	73	234	117
99	143	200	129	82	192	170	104	235	136	81	93	173
205	94	236	52	105	228	46	5	198	254	57	155	97
133	142	171	199	50	187	181	65	107	127	226	147	218
184	33	131	86	77	44	31	62	88	18	238	43	24
23	154	159	80	111	134	114	9	91	3	130	16	10
83	240	195	119	253								

Tabela A.3: Substitucijska tabela T_1

19	11	80	114	43	1	69	94	39	18	127	117	97
3	85	43	27	124	70	83	47	71	63	10	47	89
79	4	14	59	11	5	35	107	103	68	21	86	36
91	85	126	32	50	109	94	120	6	53	79	28	45
99	95	41	34	88	68	93	55	110	125	105	20	90
80	76	96	23	60	89	64	121	56	14	74	101	8
19	78	76	66	104	46	111	50	32	3	39	0	58
25	92	22	18	51	57	65	119	116	22	109	7	86
59	93	62	110	78	99	77	67	12	113	87	98	102
5	88	33	38	56	23	8	75	45	13	75	95	63
28	49	123	120	20	112	44	30	15	98	106	2	103
29	82	107	42	124	24	30	41	16	108	100	117	40
73	40	7	114	82	115	36	112	12	102	100	84	92
48	72	97	9	54	55	74	113	123	17	26	53	58
4	9	69	122	21	118	42	60	27	73	118	125	34
15	65	115	84	64	62	81	70	1	24	111	121	83
104	81	49	127	48	105	31	10	6	91	87	37	16
54	116	126	31	38	13	0	72	106	77	61	26	67
46	29	96	37	61	52	101	17	44	108	71	52	66
57	33	51	25	90	2	119	122	35				

Tabela A.4: Substitucijska tabela T_2

52	50	44	6	21	49	41	59	39	51	25	32	51	47	52	43
37	4	40	34	61	12	28	4	58	23	8	15	12	22	9	18
55	10	33	35	50	1	43	3	57	13	62	14	7	42	44	59
62	57	27	6	8	31	26	54	41	22	45	20	39	3	16	56
48	2	21	28	36	42	60	33	34	18	0	11	24	10	17	61
29	14	45	26	55	46	11	17	54	46	9	24	30	60	32	0
20	38	2	30	58	35	1	16	56	40	23	48	13	19	19	27
31	53	47	38	63	15	49	5	37	53	25	36	63	29	5	7

Tabela A.5: Substitucijska tabela T_3

1	5	29	6	25	1	18	23	17	19	0	9	24	25	6	31
28	20	24	30	4	27	3	13	15	16	14	18	4	3	8	9
20	0	12	26	21	8	28	2	29	2	15	7	11	22	14	10
17	21	12	30	26	27	16	31	11	7	13	23	10	5	22	19

Tabela A.6: Substitucijska tabela T_4

15	12	10	4	1	14	11	7	5	0	14	7	1	2	13	8
10	3	4	9	6	0	3	2	5	6	8	9	11	13	15	12